A Discrete Time Heuristic for Storage and Scheduling Unloading Operations in Container Terminal under Capacity and Non-Interference Constraints

Sanae Kouismi
MOAD-SCM Team, Mohammadia School of Engineering, Mohammed V University in Rabat, Morocco
Orcid: 0000-0001-5599-4266

Abstract
The operational management of a container terminal is critical for the terminal’s efficiency due to many constraints. The main objective of this paper is to solve a real-life optimization problem at the logistical platform MITA located in Casablanca, Morocco, namely the completion time minimization of containers unloading operations on each train. Thus, we study the cranes scheduling problem as well as the storage strategy in planning horizon while taking into account two real-life constraints which are cranes non-interference and storage areas capacity constraints. A discrete time scheduling heuristic is presented to solve the problem. Computational experiments have been conducted proving the efficiency of the proposed heuristic to find a good solution.

Keywords: Container terminal; Storage operations; Scheduling heuristic; Discrete time; Makespan; Interference.

INTRODUCTION
Containers transportation between the Tanger Med port (Morocco) and the port of Casablanca (Morocco) is provided by truck, rail or by small boats (feeders). Currently, ONCF (the National Railway Office) is facing a competition from feeders in terms of transport capacity. ONCF is seeking to increase its transport capacity on this line by maximizing the number of trains that can be treated in its MITA platform. The treatment of a train consists of containers unloading, transporting and placing in storing areas using cranes. Thus, scheduling all these operations is necessary in the management of the platform and needs to take into account the available resources and also the potential interference between cranes in storage zones. The objective is to minimize the makespan of all these operations for a train waiting at a quay. Each train contains at most 60 containers. MITA is directly connected to the national rail network and it has two quays with a total length of 1.4 km. This paper studies the problem of minimizing the makespan of all these operations given the available resources in terms of cranes and storage areas while taking into account the interference risks of cranes in storing zones. Thus, the objective of this study is to assign each container to a crane and to a storing area and to determine its starting time of treatment in order to achieve all the operations in the shortest time.

The present paper is organized as follows: Section 2 gives a brief review of previous researches in container operations terminal. In section 3, the MITA terminal and the problem are described in details. In section 4, we will present the scheduling heuristic based on the multiple insertion technique. Section 5 presents the results of computational experiments used to evaluate the algorithm. Finally, in Section 6 we present our conclusions.

LITERATURE REVIEW
Problems related to the exploitation of container terminals drew gradually more much attention and were widely studied recently because of the great importance of container terminals. In this section, we present a brief review of existing studies bound to the problem of sequencing the operations in sea or dry port. Containers are transshipped from a mode of transport to another one and this process must be characterized by quickness and efficiency. So, that the complexity of the operations of system requires the development of several methods to support decision for each sub-process suitable to the kind of used cranes. We describe herein the Scheduling and Routing of several types of cranes.

The quay crane scheduling problem consists of scheduling operations on cranes that are assigned to a ship for its service (Meisel, 2011). It was initially defined by Daganzo (1989), who proposed a MIP model for assigning cranes to bays at specific time slots by keeping the balance for the total workload between cranes. The author solved the model using exact and heuristic algorithms. Zeng, Diabat and Zhang (2015) developed a MIP model for quay crane dual-cycling movements in the terminal. The main objective of this model is executing a stowage schedule of operation sequences of containers movements and quay cranes. The authors proposed bi-level genetic algorithm as solution method.

The main function of yard vehicles is the connection between the storage area and the quay by the transport of containers for
export or import. AGVs (Automated Guided Vehicules) have the capacity to load a 40-feet container or two 20-feet containers (Steenken & Stahlbock, 2004). Kim and Bae (2004) studied the dispatching of AGVs by synchronizing the operations of the handling equipment in a dynamic environment. The proposed MIP model is solved by a heuristic algorithm whose performance was tested via simulation. Based on simulation, Liu and Ioannou (2002) compared different AGV dispatching rules in container simulation. Based on simulation, Liu and Ioannou (2002) compared different AGV dispatching rules in container terminal in order to find the suitable rule to implement. The authors developed four heuristics and discuss its performance.

The problem of managing the storage yard is treated by Lee, Chew, Tan and Han (2006). The authors tried to reduce traffic congestion by minimizing the necessary number of cranes for achieving the operating loads. The developed MIP consists of the allocation of containers to storage blocks and the assignment of YCs (Yard Cranes) to the storage blocks. Sequential method as algorithm for generating columns is proposed to solve the problem. He, Chang, Mi and Yan (2010) treated the YC scheduling problem to minimize the weighed number of delayed jobs and the interblock YC distance. They solved the problem using a hybrid heuristic, which uses rolling-horizon approach and parallel Genetic algorithm. Chang, Jiang, Yan and He (2011) developed also a dynamic rolling-horizon approach for a YC scheduling problem, which allows to obtain near optimal solutions better than those generated by the corresponding static rolling-horizon approach.

Wang and Yun (2013) proposed an approach for YTs (Yard Trucks) dispatching through a MIP model to minimize a weighted sum of total delay and total yard trucks travel times. The proposed MIP was solved using a genetic algorithm. Ng, Mak and Zhang (2007) studied the YTs scheduling problem at a container terminal. The objective was to minimize the makespan of operations. The problem was solved using a genetic algorithm. Lee, Cao and Shi (2008) developed the same type of algorithm to solve also the YTs scheduling problem and storage allocation for import containers and to minimize the makespan. But in (Lee, Cao, Shi, & Chen, 2009), the authors solve this problem using Hybrid Insertion Algorithm. The objective of this study was to minimize the delay and the total truck transportation time for two operations: loading and unloading to and from the ship.

In small or medium terminals, rubber-tired gantries (RTGs) or reach stackers (RSs) are used for loading/unloading operations (Ballis & Golias, 2002) so that the handling equipment can be more flexibly allocated to several segments of the terminal. In the literature, few research studies have been interested to the reach stacker cranes and these studies are referred to intermodal terminals rather than to seaports. Ambrosino, Bramardi, Pucciano, Sacone and Siri (2011) analyzed the problem of loading containers in the train using the RSs cranes. The authors presented two mathematical models and proposed a heuristic approach to minimize the rehandling operations of the handling equipment in a dynamic environment. The proposed MIP model is solved by a heuristic algorithm whose performance was tested via simulation. Based on simulation, Liu and Ioannou (2002) compared different AGV dispatching rules in container terminal in order to find the suitable rule to implement. The authors developed four heuristics and discuss its performance.

The problem of managing the storage yard is treated by Lee, Chew, Tan and Han (2006). The authors tried to reduce traffic congestion by minimizing the necessary number of cranes for achieving the loading operations. The developed MIP consists of the allocation of containers to storage blocks and the assignment of YCs (Yard Cranes) to the storage blocks. Sequential method as algorithm for generating columns is proposed to solve the problem. He, Chang, Mi and Yan (2010) treated the YC scheduling problem to minimize the weighed number of delayed jobs and the interblock YC distance. They solved the problem using a hybrid heuristic, which uses rolling-horizon approach and parallel Genetic algorithm. Chang, Jiang, Yan and He (2011) developed also a dynamic rolling-horizon approach for a YC scheduling problem, which allows to obtain near optimal solutions better than those generated by the corresponding static rolling-horizon approach.

Wang and Yun (2013) proposed an approach for YTs (Yard Trucks) dispatching through a MIP model to minimize a weighted sum of total delay and total yard trucks travel times. The proposed MIP was solved using a genetic algorithm. Ng, Mak and Zhang (2007) studied the YTs scheduling problem at a container terminal. The objective was to minimize the makespan of operations. The problem was solved using a genetic algorithm. Lee, Cao and Shi (2008) developed the same type of algorithm to solve also the YTs scheduling problem and storage allocation for import containers and to minimize the makespan. But in (Lee, Cao, Shi, & Chen, 2009), the authors solve this problem using Hybrid Insertion Algorithm. The objective of this study was to minimize the delay and the total truck transportation time for two operations: loading and unloading to and from the ship.

In small or medium terminals, rubber-tired gantries (RTGs) or reach stackers (RSs) are used for loading/unloading operations (Ballis & Golias, 2002) so that the handling equipment can be more flexibly allocated to several segments of the terminal. In the literature, few research studies have been interested to the reach stacker cranes and these studies are referred to intermodal terminals rather than to seaports. Ambrosino, Bramardi, Pucciano, Sacone and Siri (2011) analyzed the problem of loading containers in the train using the RSs cranes. The authors presented two mathematical models and proposed a heuristic approach to minimize the rehandling movements and maximize utilization of the train. Usually, the terminal operation process needs integrated more than one type of handling equipment. Some authors investigated the integrated scheduling approach for container terminal operations. Wu, Luo, Zhang and Dong (2013) studied the problem of scheduling different types of equipment and integrated a mixed integer linear programming model and a mixed integer nonlinear programming model. The authors presented a schedule storage policy. Karam and Eltawil (2016) were interested in QC assignment problem and the assignment of internal trucks to each QC. The authors developed a Lagrangian relaxation and subgradient optimization procedure-based heuristic to solve the proposed model. Cao, Lee, Chen and Shi (2010) integrated YT and YC scheduling model and suggested algorithms based on Benders decomposition to solve it. Chen, Langevin and Lu (2013) studied the coordination among several cranes handling and truck transportation in a maritime container terminal by integrating them simultaneously. The objective was to increase the terminal productivity. YTs were shared among several vessels, which may reduce empty truck trips in the terminal. The proposed model is solved by developing a three-stage algorithm.

In real-word, utilizing one type of cranes or more can increase the risk of interference. So, it is necessary to integrate that into support decision model. Boysen and Fliedner (2010) presented two different strategies to express non-crossing constraint. Interference between YCs may have effects on the planning of QCs. Jung, Park, Lee, Kim and Ryu (2006) presented a GRASP algorithm in order to solve the quay crane scheduling problem considering the influence of yard cranes delays on the quay cranes operations due to the risk of interference. The objective is the minimization of the makespan. In (Lee, Wang, & Miao, 2008), the authors solved a QCs Scheduling MIP model with interference by genetic algorithms. Many research works have treated the problem of multiple cranes (YCs or QCs) scheduling with interferences. Ng (2005) studied the efficiency of container treatment with interference and solved the problem of scheduling multiple yard cranes by dynamic programming-based heuristic. Cao, Lee, and Meng (2008) developed a discrete time integer model for the double rail mounted gantry crane scheduling unloading operations problem with crane interferences. The authors proposed a scheduling heuristic, a greedy heuristic and simulated annealing based approach to solve the problem.

In this paper, we propose a new scheduling heuristic based on a discrete-time approach, which allows solving the real problem in polynomial time considering cranes and spans as machines. Moreover, the type of cranes used in MITA terminal (RSs), is rarely cited in the literature (Ambrosino, Bramardi, Pucciano, & Sacone, 2011). Next section will describe the problem.
PROBLEM FORMULATION

System description of MITA terminal

In the container terminal of the MITA dry port, two trains are often docked in the two lines quay every day. Each train is unloaded by multiple reach stacker cranes and containers are stored within storage areas (Figure 1). The dispatching and scheduling of equipment and locations of storage areas are key planning problems faced by terminal planners. Our work aims to minimize the completion time of unloading all the containers in the train waiting in the line quay. The optimal plan must ensure the safety of the traffic in the dry port, guarantees the non-interference of cranes, and takes into account the capacity of terminal resources. An optimal planning, with minimal makespan of all unloading operations of train, can increase terminal throughput by increasing the number of trains unloaded. Figure 2 shows a typical partial container terminal: the storage areas are divided into multiple blocks; each yard block consists of contiguous stacks; and each stack has a certain capacity (5 containers).

Figure 1: Flow containers in MITA terminal

Figure 2: Set of stacks that represent spans

Assumptions and problem settings

In a previous formulation of the problem (Kouismi, Benabbou, & Sbihi, 2016), a MIP model has been developed to formulate the storage and cranes scheduling problem. It consists of minimizing the makespan of unloading operations of cranes that transport containers from the quay to storage areas taking into account the problem assumptions: The storage zones are considered to be divided to many blocks and spans as shown in Figure 2. The containers waiting in the quay are 20-feet size and they are transported by RSs having the same capacity and speed. The problem of interference in every area between two blocks (span) to keep at most one RS working inside each span at any time.

Mathematical model

The following notations are introduced:

- \( N \): Total number of containers to be handled in the planning horizon
- \( M \): Total number of stacks
- \( L \): Total number of available cranes in storage areas
- \( T \): Planning horizon
- \( S \): Set of spans such as \( S = \{S_1, S_2, ..., S_K\} \)
- \( S_j \): A span which is a set of stacks
- \( \text{cap}(s) \): Capacity of stack \( s \)
- \( d_{is} \): Processing time of container \( i \) from its position on the train to the span that contain the stack \( s \) when that will be affected
- \( d'_{is} \): Processing time of container \( i \) in the span that contain the stack \( s \) before and during storage
- \( d''_{is} \): Travelling time of empty crane after the storage of container \( i \) in the span that contain the stack \( s \).
- \( \beta \): Travelling time of empty crane in the quay to recuperate other container from the train
- \( Z(s) = \{s' \in M/ \text{there is a risk of interference if there exists another crane in } s\} \): the areas of interference risk. Within this paper context, we note that this area is a span that contain a certain number of adjacent stacks, so for the model, the following notation is adopted: the span \( j \) is the set \( S_j = \{s' \in M/s \text{ is an adjacent stack to } s'\} \)

The decision variables are:

- \( y_{ist} \): A binary variable that is equal to 1 if and only if container \( i \) is assigned to stack \( s \) and if its processing starts at \( t_i \)(1 \( i \leq N; 1 \leq s \leq M; 1 \leq t \leq T) \)
- \( C_{\text{max}} \): Completion time;
Mathematical model:

\[
\begin{align*}
\text{Min } & C_{\text{max}} \\
C_{\text{max}} & \geq \sum_{s=1}^{M} \sum_{t=1}^{T} y_{ist} (t + d_{is} + d'_{is}) \quad \forall 1 \leq i \leq N \quad (1) \\
\sum_{s=1}^{M} \sum_{t=1}^{T} y_{ist} & = 1, \quad \forall 1 \leq i \leq N \quad (2) \\
\sum_{i=1}^{N} \sum_{t=1}^{T} y_{ist} & \leq \text{cap}(s), \quad \forall 1 \leq s \leq M \quad (3) \\
\sum_{i=1}^{N} \sum_{s=1}^{M} \sum_{t'=\text{Max}(0,t-d_{is}(\alpha+1)+d_{is}+\beta)}^{t} y_{ist'} & \leq L, \quad \forall 1 \leq t \leq T \quad (4) \\
\sum_{i=1}^{N} \sum_{s \in S_j} \sum_{t'=\text{Max}(0,t-d_{is}(\alpha+1))}^{t} y_{ist'} & \leq 1, \quad \forall 1 \leq t \leq T, \forall S_j \subseteq S, 1 \leq j \leq K \quad (5)
\end{align*}
\]

Computational experiments are carried out using CPLEX. Eight small sized randomly test problems were solved in (Kouisi et al., 2016) under the following considerations:

- Number of cranes: 3, 4, 5
- Number of containers: 10
- Number of stacks: 16, 24

However, the optimal schedule was not founded for the real problem (number of cranes: 7, 8, 9; number of containers: 20, 40, 60; number of stacks: 24, 36, 48, 64, 80, 100, 120). As an alternative, we propose a discrete time scheduling heuristic.

**DISCRETE TIME SCHEDULING HEURISTIC**

**Constructive heuristic**

Several heuristics have been formulated to find a near-optimal solution in short time. Among the set of scheduling algorithms that try to sequence a preordered list of jobs respecting some criteria, NEH (Nawaz Enscore Ham) heuristic is introduced by Nawaz, Enscore and Ham (1983). According to Leisten (1990), this heuristic is considered as the best constructive heuristic. As described in (Khalouli, 2010), the NEH heuristic solves scheduling problems with m machines by minimizing the makespan. As depicted in Algorithm 1, the method begins by sorting, in decreasing order, jobs according to their processing time. Then, it finds a solution by successive insertions of jobs.

**Algorithm 1:** Classical NEH heuristic:

1: Sort the jobs in order of due dates

LOOP Process

2: for each job i do

3: \( \text{Makespan} \leftarrow \text{InitialMakespan} \)

4: for each machine m do

5: for each position p in machine m do

6: \( \text{Makespan}^{'} = \text{INSERT} \text{ job } i \text{ on } m \text{ at position } p \)

7: if (Makespan’ < Makespan) then

8: \( \text{Makespan} \leftarrow \text{Makespan}^{'} \)

9: end if

10: Remove i from m at position p;

11: end for

12: end for

13: end for
Discrete time scheduling heuristic

In order to respect the non-interference constraint across the planning horizon, each span \( j \) (respectively crane \( m \)) is considered as a machine with AvailableTime to ensure the order of the sequences. In addition, each container \( i \) is characterized with two measurements:

- \( \text{StartingTime}(i) \): beginning date of container \( i \);
- \( \text{CompletionTime}(i) \): completion time of the container \( i \).

We developed a discrete time scheduling heuristic, inspired from the NEH algorithm, to find the optimal sequences of containers. In one hand, we considered spans and cranes as the NEH machines. In the other hand, the stacks and the available time of spans and cranes are considered as the positions of those machines.

The developed heuristic is described in Algorithm 2. The selection of available spans and cranes is based on the following rules:

1. If there is already a crane in the span, no crane can be moved into span;
2. If all cranes are busy, containers on the train must wait until the availability of loading cranes;
3. If a sequence of the crane, span and stack give the minimum completion time of container, we choose this sequence.

We illustrate an example in Figure 3, in order to affect the container \( y \) to the Span and Crane, the algorithm tests all possibilities and the optimal one is \( S2 \) and \( C2 \), so the sequence that affected to the container \( y \) is \((s; t; C2)\) with \( s \in S2 \) and \( t = \text{Max(CompletionTime}(x1); \text{completionTime}(x2)) \).

**Figure 3:** The Optimal affectation of container \( Y \) to span \( S2 \) and crane \( C2 \)

**Algorithm 2:** Discrete Time Scheduling Heuristic

1: LOOP Process
2: for each container \( i \) do
3:     \( \text{Makespan} \leftarrow \text{InitialMakespan} \)
4:     for each span \( j \) do
5:         for each stack \( s \) in span \( j \) do
6:             for each crane \( m \) do
7:                 if \( \text{Capacity}(s) \neq 0 \) then
8:                     if no container is already affected to \( j \) then
9:                         \( \text{AvailableTime}(j) \leftarrow 0 \)
10:                    else
11:                        \( \text{AvailableTime}(j) \leftarrow \text{StartingTime(last container affected to } j) + \text{time that occupied this container the span } s \)
12:                    end if
13:                 if no container is already affected to \( m \) then
14:                     \( \text{AvailableTime}(m) \leftarrow 0 \)
15:                else
16:                    \( \text{AvailableTime}(m) \leftarrow \text{StartingTime(last container affected to } m) + \text{time that occupied this container } m + \text{return time of empty crane } m \text{ to the quay} \)
17:                end if
18:             \( \text{StartingTime}(i) \leftarrow \text{Max(AvailableTime}(j), \text{AvailableTime}(m)) \)
19:             \( \text{CompletionTime}(i) \leftarrow \text{StartingTime}(i) + \text{ProcessingTime}(i) \)
20:             if \( \text{CompletionTime}(i) < \text{Makespan} \) then
21:                 \( \text{Makespan} \leftarrow \text{CompletionTime}(i) \)
22:                 \( \text{Affect the current container } i \text{ to the stack } s \text{ in span } j \text{ and to crane } m \text{ in the } \text{StartingTime}(i) \)
23:                 \( \text{Update} \text{ Capacity}(s) \leftarrow \text{Capacity}(s) - 1 \)
24:                 if container \( i \) exist in other span \( j' \) (resp. crane \( m' \)) then
25:                     \( \text{Remove } i \text{ from } j' \text{(resp. } m') \)
26:                 \( \text{Update} \text{ Capacity}(s' \in j') \leftarrow \text{Capacity}(s' \in j') + 1 \)
27:             end if
28:         end for
29:     end for
30: end for
31: end for
32: end for
33: end for
COMPUTATIONAL RESULTS

In this section, the computational results of various experiments is presented to draw conclusions regarding the resources, as well as to evaluate the efficiency of the proposed approach.

Instances generation

Some experimental tests are presented as small instances in order to analyze and to compare the results obtained by applying the heuristic algorithm proposed in Section 4 and the exact method used by CPLEX. For each 10 containers we solved different instances. We consider that the import zone on the terminal is composed by 16 and 24 stacks. The number of cranes is varied between 3, 4 and 5. The capacity of stacks is 4 containers. We suppose that the processing times \( d_i \) and \( d'_i \) are generated using a uniform distribution. Based on the same formula using in (Kouismi et al., 2016) we calculated time horizon:

\[
T = \frac{1}{L} \left( \text{Max}(2(d_i + d'_i)) \right) \quad \forall 0 \leq i \leq N, \forall 0 \leq s \leq M \quad (7)
\]

For medium and large instances, the experimental tests have been run on a set we have ran the experimental tests on a set of randomly generated problem instances with data inspired from the case study of MITA. We consider the following: number of containers: 20, 40 and 60; number of stacks: 24, 36, 48, 64, 80, 100, and 120. We consider also various combinations of cranes, spans and stacks. The processing times are randomly generated using uniform distribution. We set \( \alpha = 0.5 \) and \( \beta = 50 \) seconds. The capacity of stack is 5 containers. For each fixed number of containers, 9 instances are generated.

Result of small sized problems

Table 1 shows the values of Cmax and computation time for different tests using the developed heuristic and CPLEX applied to the MIP. In (Kouismi et al., 2016), solving exactly the problem takes a large computational time and it requires many hours for \( N > 10 \). The results in Table 1 show that the proposed heuristic gives a near optimal solution in a short time compared to the CPU time (Computational Time) executed by CPLEX by a significant gain time.

<table>
<thead>
<tr>
<th>ID</th>
<th>N</th>
<th>M</th>
<th>L</th>
<th>K</th>
<th>Cmax(s)</th>
<th>CPU time(s)</th>
<th>Cmax(s)</th>
<th>CPU time(s)</th>
<th>Gain(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>16</td>
<td>3</td>
<td>4</td>
<td>628</td>
<td>362,972</td>
<td>676</td>
<td>0.094</td>
<td>362,878</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>16</td>
<td>4</td>
<td>4</td>
<td>452</td>
<td>356,943</td>
<td>502</td>
<td>0.093</td>
<td>356,850</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>16</td>
<td>5</td>
<td>4</td>
<td>368</td>
<td>505,813</td>
<td>421</td>
<td>0.078</td>
<td>505,735</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>24</td>
<td>3</td>
<td>6</td>
<td>623</td>
<td>294,350</td>
<td>665</td>
<td>0.093</td>
<td>294,257</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>24</td>
<td>4</td>
<td>6</td>
<td>457</td>
<td>598,377</td>
<td>495</td>
<td>0.125</td>
<td>598,252</td>
</tr>
</tbody>
</table>

Result of medium and large sized problems

In this section, we will test the scheduling heuristic in a set of medium and large sized problems. The total number of containers ranges from 20 to 60. As illustrated in Table 2, for each range, we find that the set of containers can be stored in an interval of time according to the resources (cranes, stacks). In fact, for 20 containers the process lasts between 9 and 13 minutes. The makespan is from 16 to 23 minutes for 40 containers. For 60 containers, the heuristic finds that completion time of all operations is between 24 and 32 minutes. The algorithm takes a reasonable computational time. For 20 containers the scheduling heuristic finds the solution without exceeding 615 ms (milliseconds). For 40 containers, according to cranes, spans and stacks, the computational time is between 37 and 260 seconds. For 60 containers, CPU time is between 2 and 6 minutes.
Table 2: Results for medium and large sized problems solved using the scheduling heuristic

<table>
<thead>
<tr>
<th>ID</th>
<th>N</th>
<th>M</th>
<th>L</th>
<th>K</th>
<th>Cmax(s)</th>
<th>CPU time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>24</td>
<td>7</td>
<td>4</td>
<td>780</td>
<td>0,560</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>24</td>
<td>8</td>
<td>4</td>
<td>725</td>
<td>0,349</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>24</td>
<td>9</td>
<td>4</td>
<td>734</td>
<td>0,382</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>36</td>
<td>7</td>
<td>6</td>
<td>693</td>
<td>0,420</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>36</td>
<td>8</td>
<td>6</td>
<td>597</td>
<td>0,297</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>36</td>
<td>9</td>
<td>6</td>
<td>587</td>
<td>0,615</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>48</td>
<td>7</td>
<td>6</td>
<td>641</td>
<td>0,453</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>48</td>
<td>8</td>
<td>6</td>
<td>576</td>
<td>0,492</td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td>48</td>
<td>9</td>
<td>6</td>
<td>560</td>
<td>0,568</td>
</tr>
<tr>
<td>10</td>
<td>40</td>
<td>48</td>
<td>7</td>
<td>6</td>
<td>1388</td>
<td>2,041</td>
</tr>
<tr>
<td>11</td>
<td>40</td>
<td>48</td>
<td>8</td>
<td>6</td>
<td>1199</td>
<td>15,633</td>
</tr>
<tr>
<td>12</td>
<td>40</td>
<td>48</td>
<td>9</td>
<td>6</td>
<td>1061</td>
<td>1,544</td>
</tr>
<tr>
<td>13</td>
<td>40</td>
<td>64</td>
<td>7</td>
<td>8</td>
<td>1302</td>
<td>2,917</td>
</tr>
<tr>
<td>14</td>
<td>40</td>
<td>64</td>
<td>8</td>
<td>8</td>
<td>1171</td>
<td>2,895</td>
</tr>
<tr>
<td>15</td>
<td>40</td>
<td>64</td>
<td>9</td>
<td>8</td>
<td>1008</td>
<td>2,856</td>
</tr>
<tr>
<td>16</td>
<td>40</td>
<td>80</td>
<td>7</td>
<td>8</td>
<td>1292</td>
<td>5,826</td>
</tr>
<tr>
<td>17</td>
<td>40</td>
<td>80</td>
<td>8</td>
<td>8</td>
<td>1154</td>
<td>2,927</td>
</tr>
<tr>
<td>18</td>
<td>40</td>
<td>80</td>
<td>9</td>
<td>8</td>
<td>989</td>
<td>2,258</td>
</tr>
<tr>
<td>19</td>
<td>60</td>
<td>80</td>
<td>7</td>
<td>8</td>
<td>1932</td>
<td>10,478</td>
</tr>
<tr>
<td>20</td>
<td>60</td>
<td>80</td>
<td>8</td>
<td>8</td>
<td>1702</td>
<td>11,042</td>
</tr>
<tr>
<td>21</td>
<td>60</td>
<td>80</td>
<td>9</td>
<td>8</td>
<td>1512</td>
<td>8,186</td>
</tr>
<tr>
<td>22</td>
<td>60</td>
<td>100</td>
<td>7</td>
<td>10</td>
<td>1923</td>
<td>13,105</td>
</tr>
<tr>
<td>23</td>
<td>60</td>
<td>100</td>
<td>8</td>
<td>10</td>
<td>1650</td>
<td>7,988</td>
</tr>
<tr>
<td>24</td>
<td>60</td>
<td>100</td>
<td>9</td>
<td>10</td>
<td>1466</td>
<td>9,191</td>
</tr>
<tr>
<td>25</td>
<td>60</td>
<td>120</td>
<td>7</td>
<td>10</td>
<td>1916</td>
<td>22,332</td>
</tr>
<tr>
<td>26</td>
<td>60</td>
<td>120</td>
<td>8</td>
<td>10</td>
<td>1648</td>
<td>19,200</td>
</tr>
<tr>
<td>27</td>
<td>60</td>
<td>120</td>
<td>9</td>
<td>10</td>
<td>1437</td>
<td>12,414</td>
</tr>
</tbody>
</table>

Performance analysis

To discuss the results, we evaluate the makespan according to the resources as shown in the Figures 4, 5 and 6; for each range of NxM, the makespan value decreases when increasing the number of cranes (respectively the number of spans) for all medium and large instances tested. A short objective value reveals a good exploitation of the cranes and spans, as illustrated in observation 1 and 2 in the Table 3.

Figure 4: Evolution of the makespan according to the number of cranes and spans for 20 containers

Figure 5: Evolution of the makespan according to the number of cranes and number of spans for 40 containers

Figure 6: Evolution of the makespan according to the number of cranes and number of spans for 60 containers
Table 3: Impact of parameters on makespan

<table>
<thead>
<tr>
<th>ID</th>
<th>N</th>
<th>M</th>
<th>Number of stacks by span</th>
<th>K</th>
<th>L</th>
<th>Cmax(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fixed</td>
<td>Fixed</td>
<td>Fixed</td>
<td>Fixed</td>
<td>587 → 641</td>
<td>641</td>
</tr>
<tr>
<td>2</td>
<td>Fixed</td>
<td>8</td>
<td>8 → 10</td>
<td>7</td>
<td>693 → 1292</td>
<td>1292</td>
</tr>
<tr>
<td>3</td>
<td>Fixed</td>
<td>6</td>
<td>6 → 8</td>
<td>7</td>
<td>1008 → 989</td>
<td>989</td>
</tr>
</tbody>
</table>

In addition, the makespan decreases according to the number of stacks existing in the span, total number of stacks M and total number of containers N. For the third remark (in Table 3), we note that the makespan decreases if we fix the number of containers and the number of cranes then we increase the number of spans and the number of stacks per span.

For example, for 20 containers and 7 cranes (respectively 8 and 9), if we make M vary from 4 to 6 and the number of stacks by span from 6 to 8, we notice that the makespan decreases from 780 s (seconds) to 641 s (respectively 725 s to 576 s and 734 s to 560 s). Also, for 40 containers and 7 cranes (respectively 8 and 9), if we make M vary from 6 to 8 and the number of stacks per span from 8 to 10, we notice that the makespan decreases from 1388 s to 1292 s (respectively from 1199 to 1154 and 1061 to 989). Finally, for 60 containers and a number of cranes fixed to 7 (respectively 8 and 9), if we increase M from 4 to 6 and the number of stacks per span from 6 to 8, the makespan decreases from 1932 s to 1916 s (respectively 1702 s to 1648 s and 1512 s to 1437 s). In addition, in the fourth observation, we fix N, M, L and we make the number of stacks per span vary, for the different range of containers 20, 40, 60, the outcomes mentioned in Table 4 are observed.

Table 4: Impact of stacks number by span

<table>
<thead>
<tr>
<th>ID</th>
<th>N</th>
<th>M</th>
<th>Number of stacks by span</th>
<th>K</th>
<th>L</th>
<th>Cmax(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>6</td>
<td>6 → 8</td>
<td>36</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>8</td>
<td>8 → 10</td>
<td>64</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>60</td>
<td>10</td>
<td>10 → 12</td>
<td>100</td>
<td>120</td>
<td></td>
</tr>
</tbody>
</table>

CONCLUSION

The non-crossing and capacitated constraints are very important to improve the productivity of unloading operations in a container terminal. In this paper, the problem of scheduling and storage operations for unloading train in a dry port has been investigated.

The objective is to minimize makespan of all unloading operations of a train. A discrete time heuristic has been developed by considering constraints of non-interference between cranes and storage areas capacity. A new scheduling constructive heuristic has been proposed to find near-optimal solutions for the problem. The original idea within this approach consists of considering spans and cranes as machines, and the stacks and the available time of spans and cranes as the positions of those machines. It has been proved that the heuristic performs well for small sized problems. Furthermore, the experimental results reported above show the efficiency of the heuristic approach; particularly for real applications. The real considered instances consist of a train composed of 60 containers and a storage area with a capacity of 100 stacks. The results of this work seems to be plausibly relevant for the MITA terminal since we have an optimal scheduling of unloading operations in very short periods of time, and it proves that MITA can increase the number of trains per day.

REFERENCES


for loading outbound containers in container terminals. *International Journal of Production Economics, 115*(1), 221–228. doi:10.1016/j.ijpe.2008.05.014


